A VLSI Architecture for H.264/AVC Variable Block Size Motion Estimation

Dam. Minh Tung and Tran. Le Thang Dong
Center of Electrical Engineering, Duy Tan University, Da Nang, Viet Nam
Email: minhtungdam@gmail.com, tranthangdong@duytan.edu.vn

***Abstract*—In this paper, we propose an efficient VLSI architecture for variable block size motion estimation (VBSME) in H.264/AVC to reduce the hardware cost and latency. The proposed architecture adopts four modes (8x8, 8x16, 16x8 and 16x16 modes) instead of seven modes for VBSME specified in H.264/AVC. Our architecture significantly reduces the hardware size by reducing (1) the registers and adders in each processing unit, (2) the comparison elements, and (3) the registers used to store the minimum SADs and motion vectors. The experimental result shows that our proposed architecture reduces the hardware size by 44.3% while it also increases the operation clock frequency by 54.9% compared with the best-known architecture. The proposed architecture satisfied the real-time processing requirement of the massive data in high resolution video applications.**

***Index Terms*—H264/AVC, VBSME, motion estimation, 1-D tree architecture, VLSI design, video codec**

I. Introduction

Block matching algorithm (BMA) is a well-known method for motion estimation widely used to reduce the temporal redundancy between successive image frames in digital video processing. For previous video compression standards such as MPEG-2, a fixed- size BMA was mostly used. In a typical BMA, each frame of a video sequence is divided into a fixed number of non-overlapping square blocks. For each block in the current frame, the best matching block is searched in the previous frame under a certain criterion. In most BMAs, the matching criterion used to produce an error cost function is the sum of absolute differences (SADs) between the 16x16 macroblocks (MBs). If *x(i, j)* and *y(i, j)* are the pixels of the relevant current and candidate MBs, and *m* and *n* are the coordinates of the motion vector (MV), the SAD is then defined as:

$$SAD(m,n) = \sum_{i=0}^{15}\sum_{j=0}^{15} | x(i, j) - y(i+m, j+n) |$$

A fixed 16x16 MB size is well suited for large areas of consistent motion, but not suitable to accommodate the different changes in object movement within a video frame. Therefore, it may limit the performance of the BMA for low bit rate video coding applications. H.264/AVC remedies this limitation by using variable block size motion estimation (VBSME). The VBSME has 41 different sub-blocks of seven modes including 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4 modes as shown in Fig. 1.

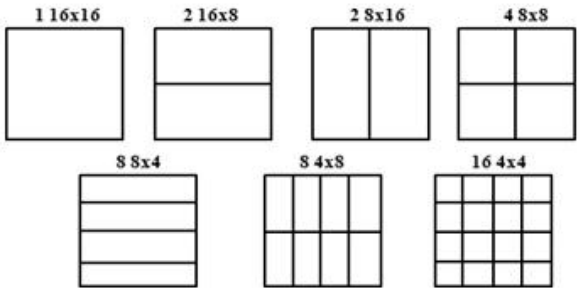


Figure 1. Variable block sizes supported by H.264/AVC

Various VLSI architectures have been proposed for VBSME implementation. Most of them are based on the traditional full search algorithm because of its good performance and regularity. However, full search algorithm requires high computational cost. Therefore, it is only effective when implemented as a dedicated hardware. Several fast algorithms such as the three step search, the new three step search, the diamond search, and the hexagon-based search have been proposed for improving motion estimation implementation. Some other advanced algorithms such as the cross-diamond search, the hardware-oriented modified diamond search [1] and the line diamond parallel search (LDPS) [2] have been proposed to provide better adaptability and searching efficiency for tracking large motions. Among these algorithms, LDPS algorithm [2] provides the better objective quality and fast searching compared with others. Furthermore, the algorithm is simple and suitable for hardware implementation. Our proposed architecture is based on one of the three widely used architectures: the Propagate Partial SAD, the SAD tree and the Parallel Sub-Tree. The SAD tree architecture is chosen due to its highest performance among the three. In this paper, we propose a new SAD tree architecture adopting four modes instead of seven modes for VBSME. LDPS algorithm is employed for motion estimation to achieve low hardware cost and high processing speed.

This paper is organized as follows. In Section 2, the related works are explained. In Section 3, the proposed architecture is explained in detail. In Section 4,

Manuscript received December 7, 2013; revised February 10, 2014.

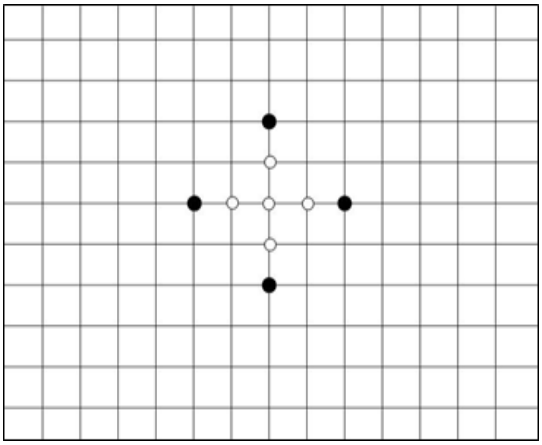
doi: 10.12720/joace.3.1.51-55

experimental results are provided. Section 5 concludes the paper.

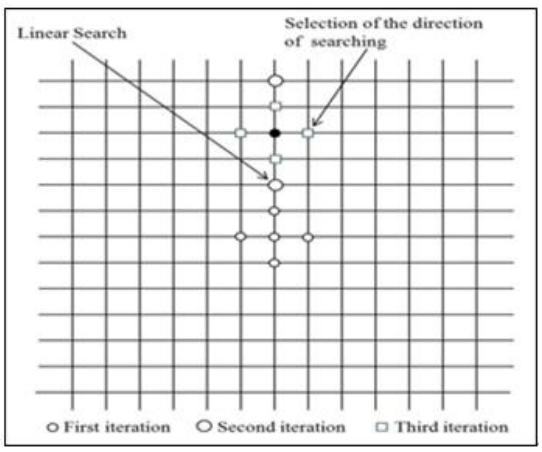
II. Related Work

A. LDPS Algorithm

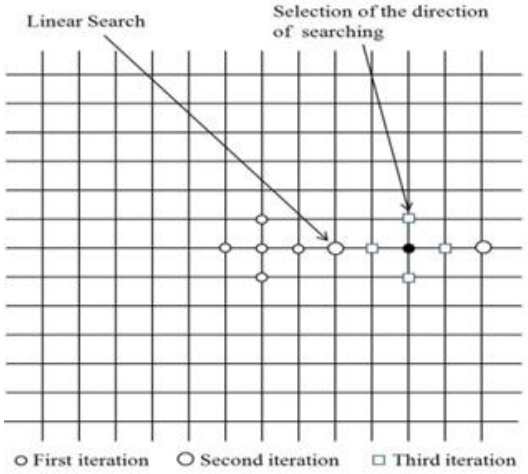
The LDPS algorithm exploits the center-biased characteristic of the real world video sequences. Fig. 2 is used to explain the LDPS algorithm. There are two patterns to search the matching point in LDPS. The first is the small diamond search pattern (SDSP) used at the beginning of a search as shown in Fig. 2(a). The second pattern improves the search in the horizontal and vertical direction as shown in Fig. 2(b) and (c). The LDPS algorithm uses the small diamond search pattern to consist of 5 points. The center point of SDSP is the center of the search window and four neighboring points are on the left, right, top and bottom of the center point as described in [2]. The checking points of the SDSP are examined one by one from the center to the outside. The SDSP determines the direction of the search either horizontal or vertical. The search continues until the minimum SAD is found at the center point of SDSP. If the minimum SAD is found at the four neighboring points around the center point, three new points are added along the chosen direction. Then, the search process is iteratively performed on the new center point. Fig. 2(d) shows the flow chart of the LDPS algorithm.



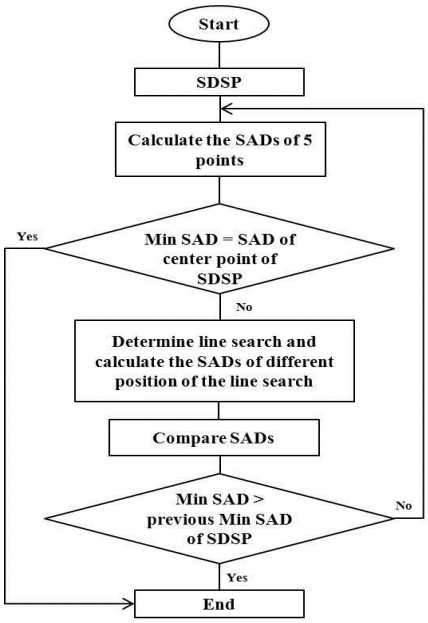
(a) The small diamond search pattern (SDSP)



(b) Vertical line search



(c) Horizontal line search



(d) Flow chart of the LDPS algorithm

Figure 2. The LDPS algorithm

B. Overview of VBSME Architecture

Since motion estimation (ME) is the most computation-intensive part in video coding process, various architectures have been proposed. For example, a 1-D and a 2-D ME architectures are proposed in [3] and [4], respectively. However, these architectures only support 16x16 mode. They are not suitable for H.264/AVC. Another 1-D architecture [1] and a 2-D architecture [5] are presented supporting the block modes in H.264/AVC. Compared with the 1-D architecture, 2-D architecture has the higher performance, but it occupies larger area.

Recently, Huang [5] has shown that the three small modes of VBSME such as 8x4, 4x8 and 4x4 can be ruled out because of their insignificant contribution to the video quality especially for high-resolution applications such as HDTV. As a result, the architecture for the four-mode VBSME can be simplified by reducing the computing resources. The 2-D architecture with the mode reduction

is presented in [5]. In this paper, we focus on the improvement of 1-D architecture by reducing the hardware size and increasing the clock frequency.

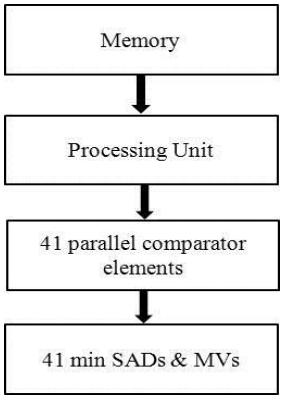


Figure 3. The general VBSME architecture

Fig. 3 shows the general architecture to implement seven modes VBSME in H.264/AVC. It needs 41 comparator elements (CEs) to compare 41 SAD values of seven modes (1 16x16, 2 8x16, 2 16x8, 4 8x8, 8 4x8, 8 8x4 and 16 4x4). However, by adopting the four modes (8x8, 8x16, 16x8 and 16x16) only, the number of CEs can be reduced from 41 to 9. Similarly, the number of register pairs to store the minimum SADs and the corresponding motion vectors can be reduced from 41 to 9. We propose a new VBSME architecture with the reduced hardware size and the improved performance in the following section.

III. Proposed Architecture

A. Four-Mode VBSME Architecture

Fig. 4 shows the overall architecture to implement the four-mode VBSME for LDPS algorithm. It consists of four main elements: (1) search window memory and current MB memory, (2) a block of processing units (PUs), (3) a comparison unit, and (4) a register storing the 9 minimum SADs and their associated motion vectors. Our architecture uses the LDPS algorithm, and hence 5 PUs are required. Each PU computes a total of 9 SADs (4 8x8 SADs, 2 8x16 SADs, 2 16x8 SADs and a 16x16 SAD) per candidate MB. Therefore, the comparison unit is composed of 9 comparison elements, which consists of one comparator and two registers. The first register stores the minimum SAD after comparison. The other stores the motion vector when the input SAD is less than the previous minimum SAD. Each comparison element processes 5 SADs of the same sub-block size received from the 5 PUs. After these SADs are compared in the comparison unit, the minimum SADs and their associated MVs are saved in the last register. We choose the first 5 search points in the LDPS algorithm and perform the computation based on the schedule shown in Table I. Then, we move to the next search point and perform the computation until we reach the best match. Table I shows that each PU is used for the SAD computation between a candidate and the current block. Hence, 5 block-matching operations can be performed concurrently in the architecture. For example, at the first clock cycle, one row of 16 pixels is calculated simultaneously in each PU. At the 16th clock cycle, each PU finishes the SAD computation.

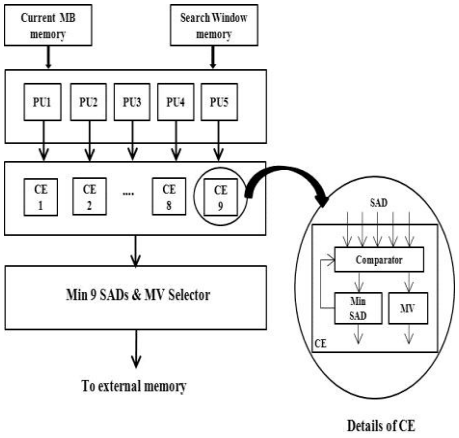


Figure 4. Four-mode VBSME architecture

TABLE I. The Computation Schedule for PU

Clock	PU1	PU2	PU3	PU4	PU5
1	$\sum_{i=0}^{15}\lvert C(i,0)-R(i,0)\rvert$	$\sum_{i=0}^{15}\lvert C(i,0)-R(i,1)\rvert$	$\sum_{i=0}^{15}\lvert C(i,0)-R(i,-1)\rvert$	$\sum_{i=0}^{15}\lvert C(i,0)-R(i+1,0)\rvert$	$\sum_{i=0}^{15}\lvert C(i,0)-R(i-1,0)\rvert$
2	$\sum_{i=0}^{15}\lvert C(i,1)-R(i,1)\rvert$	$\sum_{i=0}^{15}\lvert C(i,1)-R(i,2)\rvert$	$\sum_{i=0}^{15}\lvert C(i,1)-R(i,0)\rvert$	$\sum_{i=0}^{15}\lvert C(i,1)-R(i+1,1)\rvert$	$\sum_{i=0}^{15}\lvert C(i,0)-R(i-1,0)\rvert$
...	...	...	...	...	...
15	$\sum_{i=0}^{15}\lvert C(i,14)-R(i,14)\rvert$	$\sum_{i=0}^{15}\lvert C(i,14)-R(i,15)\rvert$	$\sum_{i=0}^{15}\lvert C(i,14)-R(i,13)\rvert$	$\sum_{i=0}^{15}\lvert C(i,14)-R(i+1,14)\rvert$	$\sum_{i=0}^{15}\lvert C(i,14)-R(i-1,14)\rvert$
16	$\sum_{i=0}^{15}\lvert C(i,15)-R(i,15)\rvert$	$\sum_{i=0}^{15}\lvert C(i,15)-R(i,16)\rvert$	$\sum_{i=0}^{15}\lvert C(i,15)-R(i,14)\rvert$	$\sum_{i=0}^{15}\lvert C(i,15)-R(i+1,15)\rvert$	$\sum_{i=0}^{15}\lvert C(i,15)-R(i-1,15)\rvert$

B. PU Architecture

Our architecture is based on the traditional 1-D tree architecture. Fig. 5 shows the structure of a PU to contain 16 processing elements (PEs). Each PE computes the absolute difference between two pixels, one from the candidate MB and the other from the current MB. PU is designed as a three-stage pipeline to simultaneously compute SADs for 16 pixels.

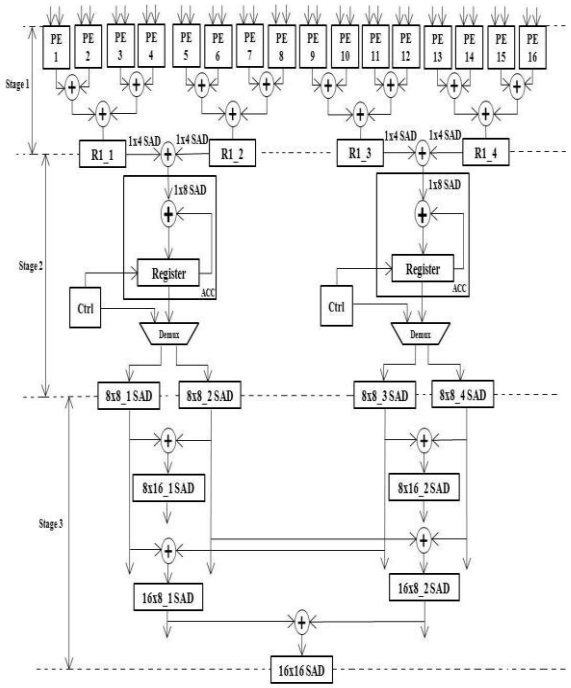


Figure 5. Architecture of a processing unit (PU)

In the first stage, 16 pixels are fed in from one row in the current MB and another row in the search area. The partial SADs of the neighboring four pixels in the same row are calculated and latched to the next adder. Finally, 1x4 SADs are stored in the registers R1_1, R1_2, R1_3, and R1_4.

In the second stage, four 8x8 SADs in one MB are calculated. Two 1x4 SADs are added to form a 1x8 SAD. Then, 1x8 SADs are accumulated in the accumulator (ACC) to form a 8x8 SAD. In addition, two 8x8 SADs are stored in the 8x8_1 SAD and 8x8_2 SAD registers after the first eight cycles. Finally, two other 8x8 SADs are stored in 8x8_3 SAD and 8x8_4 SAD registers after the next eight cycles.

In the third stage, four 8x8 SADs are input to calculate 16x8, 8x16 and 16x16 SADs. There are four new 8x8 SADs generated every 16 cycles. At the 17th cycle, 8x16_1 SAD is computed by adding 8x8_1 SAD and 8x8_2 SAD. The same operations are performed to calculate 8x16_2 SAD, 16x8_1 SAD and 16x8_2 SAD. At the 18th cycle, 16x16 SAD is computed by adding 16x8_1 SAD and 16x8_2 SAD.

Thus, at every clock cycle, the proposed pipelined architecture computes one new SAD value and stores the minimum SAD value. Since our architecture is proposed for VBSME with four modes instead of seven modes, the hardware size is significantly reduced by saving 737-bit registers and 32 adders all together.

Furthermore, the number of accumulators is reduced from sixteen to two compared with the architecture in [1] by moving each accumulator from a PE to the new location in stage-2 as shown in Fig. 5. Our architecture also saves the gate count by decreasing the bit-length of each adder. For example, an adder at the first stage of PU has only 8-bit instead of 15-bit in [1].

IV. Experimental Results

Our proposed architecture is implemented in TSMC 0.18μm process. The hardware size is reduced and the clock frequency is increased. The hardware size of our design is compared with that of the best-known 1-D tree architecture [1] in Table II. The overall size is reduced from 268K gates to 149.2K gates by 44.3%. If we rule out the common memory buffers and compare the size of the logic gates only, it is reduced from 182K gates to 63.2K gates by 65.3%.

TABLE II. Hardware Size Comparison

	Hardware size (K Gates)	
	[1]	Proposed Architecture
PU	16.93	10.54
5PUs	84.65	52.71
Current MB	14.3	14.3
SW buffer	71.7	71.7
Other	97.35	10.49
Total	268	149.2

The size of each PU in our architecture is 10.542K gates. Compared with 16.93K gates of the previous architecture [1], the hardware size is reduced by 37.7%. Moreover, our proposed architecture only uses 9 CEs and 9 register pairs instead of 41 CEs and 41 register pairs, respectively. Therefore, the hardware size in these two parts of our architecture is significantly reduced by 89.2% compared with that in the previous architecture [1].

TABLE III. Comparison of Our Proposed Architecture with Previous Design

Design	Proposed Architecture	[1]	[2]	[5]	[6]	[7]
PE number	16	16	16	256	16	16
Technology	0.18μm	0.13FPGA	0.18μm	0.18μm	0.18μm	0.18 μm
Frequency(MHz)	546.4	246.5	100	227	48.67	50
Gate count (K)	149.2	268	157	466	546	301
Block size	16x16 to 8x8	16x16 to 4x4	16x16	16x16 to 8x8	16x16	16x16 and 8x8

Table III shows the comparison between our proposed architecture and five other previous architectures. All the architectures in comparison have the same I/O size. Therefore, the operating clock frequency shows their

performance. Each architecture is implemented by different technology with various parameters. Therefore, it is difficult to compare them directly. However, the comparison still gives us meaningful information. Compared with 2-D architecture in [5], our proposed architecture has smaller size and higher frequency. The 1-D tree architecture in [1] is the best-known architecture so far.

However, it only achieves the second highest frequency and the third smallest hardware size in Table III. Furthermore, their architecture requires more registers and adders to calculate the sub-blocks of the VBSME such as 4x4, 4x8 and 8x4 modes in H.264/AVC. The implementation in [2] adopts LDPS algorithm and achieves the second smallest hardware size. However, the block size supported is only 16x16, which leads to poor image quality. Moreover, their operating clock frequency is very low (100 MHz) compared with others.

The architectures in [6], [7] are also 1-D tree architectures. However, they require the largest hardware size and the lowest clock frequency compared with others in Table III. Besides, 16x16 mode is only supported in [6] and 16x16 mode and 8x8 mode are only supported in [7], which leads to poor image quality. Our proposed architecture has the smallest gate count of 149.2 K gates. Furthermore, it achieves the highest frequency of 546.4 MHz compared with other architectures shown in Table III.

V. Conclusion

In this paper, we presented new 1-D tree architecture for VBSME. By adopting four modes instead of seven modes for VBSME, the hardware size of our architecture was significantly reduced. The numbers of CEs and the register pairs are reduced from 41 to 9, respectively. Moreover, the accumulator was modified to add the sum of absolute differences for every eight cycles. Compared with other architectures known so far, we achieved the highest operating clock frequency and the smallest hardware size. Compared with the best-known 1-D tree architecture in [1], our architecture decreased the hardware size by 44.3% and increased the clock frequency by 54.9%. For applications with massive amount of data such as HDTV and 3DTV, the area and speed of our design are the best compared with the designs known so far.

References

[1] O. Ndili and T. Ogunfunmi, "Algorithm and architecture co-design of hardware-oriendted, modified diamond search for fast motion estimation in H.264/AVC," *IEEE Trans. Circuits Syst.Video Technol*., vol. 21, no. 97, 2011.

[2] M. Kthiri, H. Loukil, I. Werda, A. Ben Atitallah, A. Samet, and N. Masmoudi, "Hardware implementation of fast block matching algorithm in FGPA for H.264/AVC," *International Multi-Conference on Systems, Signals & Devices*, 2009.

[3] P. M. Kuhn, "Fast MPEG-4 motion estimation: Processor based and flexible VLSI implementations," *J. VLSI Signal Process*., vol. 23, pp. 67-92, 1999.

[4] C. H. Chou and Y. C. Chen, "A VLSI architecture for real-time and flexible image template matching," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, 1989.

[5] Y. Huang, Z. Liu, Y. Song, S. Goto, and T. Ikenaga, "Parallel improved hdtv720p targeted propagate partial sad architecture for variable block size motion estimation in h.264/avc," *IEICE Trans. on Fundamentals*, vol. E91-A, no. 4, pp. 987-997, 2008.

[6] W.-M. Chao, C.-W. Hsu, Y.-C. Chang, and L.-G. Chen, "A novel search," in *Proc. IEEE Int. Symp. Circuits Syst*., vol. 2, pp. II-492–II- 495, 2002.

[7] S.-S. Lin, "Low-power motion estimation processors for mobile video application," M.S. thesis, Graduate Inst. Electron. Eng., Nat. Taiwan Univ., Taipei, Taiwan, 2004.

Dam Minh Tung received B.S degree in Electronic and Telecommunication Engineering from Da Nang university, Viet Nam in 2010, and M.S.degree in Information and Communication Engineering from Myongji University, Korea, in 2013. He is currently a researcher at Center of Electrical Engineering (CEE), Duy Tan University, Da Nang City, Viet Nam. His research interests include VLSI and FPGA design, hardware architecture and processors.

Tran Le Thang Dong received B.S degree Electronics and Telecommunication Bachelor from Duy Tan university, Viet Nam in 2009, and M.S.degree in Computer Science from Duy Tan University, Viet Nam, in 2012. He is currently a Director at Center of Electrical Engineering (CEE), Duy Tan University, Da Nang City, Viet Nam. His research interests include image processing, design automation of embedded systems, FPGA design.